Approximation Algorithm

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 11

Lecture 54

Lecture 54 : Approximation Algorithm for Multicut Cont.

So, in the last lecture we have started seeing the multicut problem, we have written down the LP relaxation of it and we have observed that it has although it has an exponentially many constraints, we can design a polynomial time separation oracle based on standard shortest path algorithms like Dijkstra. which enables us to get an optimal solution for that linear programming relaxation using ellipsoid method for example. and then using an optimal solution we defined a metric which is also called shortest path metric on a graph. And using those metrics we defined balls around some vertices s_i of radius r and then we discuss that it will be useful to view each edge as a pipe of length x_e and cross sectional area of c_e. And hence the contribution of this edge e to the objective function is the volume of the pipe.

and using this we have then defined the volume of all pipes in the radius of ball are around s_i plus $\frac{V^*}{k}$ and then we stop. So, let us begin from there and see how that quantities are useful to design $O(\log k)$ factor approximation algorithm ok. So, let us begin multicut problem continue. So, let us recall we had defined volume of ball centered at s_i of radius r to be $\frac{V^*}{k}$ plus the volume of all pipes that are at most distance r from s_i.

So, there are two kinds of pipes which are correspond to edges of the graph where both end points belong to the ball. u and v both belong to the ball B centered at s_i of radius r $c_e x_e^*$ plus there are balls whose one end point exactly one end point there are edges whose exactly one end point belongs to the ball u belongs to $B(s_i, r)$, but v does not belong to $B(s_i, r)$. So, what is the contribution here? c_e times r minus distance of $s_i$and u. Now, the what is the role of first term? The first term ensures that $V(s_i, r)$ is nonzero irrespective of the value of r.

The first term ensures $\frac{V^*}{k}$ ensures that $V(s_i, r)$ is not equal to 0 irrespective of r in particular $V(s_i, 0)$ is equal to $\frac{V^*}{k}$ which is not equal to 0. And, the second term and the second and third term if we add over all edges, then we will get from here the it will get a connection to the LP-opt ok. Now, let us define some more notation. which we have used before also for a given radius r. Let look at the ball of radius r from s_i centered at s_i.

These are the set of vertices whose distance is at most r from s_i. So, these are subset of vertices. So, for any subset of vertices the notion of boundary edges make sense. So, δ of this are the boundary edges of this subset recall what are boundary edges of a subset these are the set of edges whose exactly one end point belongs to the subset. And I want to compute the sum of costs of this edges.

So, see this is the sum of the costs of edges whose exactly one end point belongs to the ball of radius r around s_i ok. Now, first we claim that there will always exist a radius r whose such that the cost of boundary edges is small in some sense. So, here is a lemma which we will use now and we will prove later. For any s_i one can find in polynomial time a radius r strictly less than half such that If I look at the ball of radius r around s_i and look at the boundary edges and their cost sum of costs, this is less than equal to $2\ln(k+1)$ $V(s_i, r)$ ok. So, this we use in our algorithm.

So, with this black box lemma let us first see the algorithm. So, we begin with the empty set F of edges, we iterate for k iterations, the goal of iteration i is to add edges in F such that s_i and t_i are disconnected in $G(V, E \setminus F)$ ok. And how do I find this edges? We use this lemma to do that we compute an less than half such that cost of boundary edges of the $B(s_i, r)$ is less than equal to $2\ln(k+1)$ $V(s_i, r)$ ok. So, once we get such an r we add the boundary edges of this ball.

to F. Notice that if I remove the boundary edges s_i and t_i gets disconnected because s_i belongs to the ball, but t_i does not belong to the ball because s_i the distance between s_i and t_i is at least 1 and this radius r is less than half. But after this we remove all the vertices of $B(s_i, r)$ along with its incident edges. from the graph and continue. So, we define this balls and boundary edges are with respect to the current set of edges in the graph. So, note that this quantity is say $B(s_i, r)$ and $V(s_i, r)$ are taken or defined with respect to the vertices and edges.

that remain in the current graph ok. So, let us write down the pseudo code of the algorithm let x^* be an optimal LP solution initial we initialize F the set of edges to be

finally, removed to be empty set $i=1,\ldots,k$, if s_i and t_i are connected in $(V, E\setminus F)$. we choose the radius r, let r less than half be such that cost of boundary edges of the ball around s_i of radius r is less than equal to $2\ln(k+1)\ V(s_i, r)$. We use the lemma to choose such an r, then we add the boundary edges of the ball of radius r around s_i in my solution that I am building $F\cup\delta(B(s_i, r))$ and then we remove the entire ball $B(s_i, r)$ and its incident edges. Remove $B(s_i, r)$ along with edges. So, this ends the if body and the for loop and then we return F. So, this is the pseudo code of the algorithm, a priori it is not at all clear that F is a valid solution, if I remove F all s_i and t_i gets disconnected. So, let us first prove this lemma our algorithm a feasible solution for the multicut problem.

So, let us see how it can go wrong the only possible way. So, let us write. So, the only possible way it can go wrong is that say at in iteration i, I am looking at the ball of radius r around s i. and I take this boundary edges in my solution, but there are and I delete these things s delete the vertices in the ball and what if this ball has an s_j - t_j. then I am not picking any edge on in the path if there is a path between s_i and t_j and of course, there is a path because both are connected to s_i then I am not removing any I am not removing the path between s_j and t_j because I am picking only boundary edges of the ball.

So, this is the only way that the algorithm can go wrong and let us argue that it cannot happen. In the sense that in a ball of radius r around s_i cannot contain both s_j and t_j for any index j. So, let us write the only possible way that F might not be a solution to the multicut problem. is that there exists an index $j\in[k]$ such that both s_j and t_j belongs to $B(s_i, r)$ ball of radius r around s_i for some index $i\in[k]$. We will argue that this cannot happen, let us see why it cannot happen.

So, if for contradiction both s_j and t_j belong to the ball of radius r around S i, then you see the distance between $s_j$and t_j. by triangle inequality this is less than equal to distance between $s_j$and s_i plus distance between s_i and T i. Now, the first one is less than equal to r because s_j is in the ball of radius r around s_i. The second one is also less than r because t_j is in the ball of radius r around s_i, but r is strictly less than half. So, this is less than equal to sorry less than strictly less than since r is strictly less than half.

But then there is a contradiction to the fact that x^* is a valid solution. In any valid solution the distance between any pair of vertices s_j and t_j must be at least 1. So, this contradicts the fact that x^* is a valid solution of the LP. So, it cannot happen that for some index j both s_j and t_j belongs to the ball. Hence, if is a valid solution of the multicut problem.

Using this what we will show is that the approximation guarantee approximation ratio of our algorithm is at most $4\ln(k+1)$. So, the theorem the approximation factor of our algorithm is at most $4\ln(k+1)$. So, in the next class we will see the proof of this theorem along with the proof of the lemma that we have used in our algorithm crucially ok. So, let us stop here. Thank you.